LA-UR-17-28475

Title: Ristra Milestone Review

Author(s): Herring, Angela M.
Dilts, Gary Allen
Ferenbaugh, Charles Roger
Garimella, Rao Veerabhadra
Kikinzon, Evgeny
Malone, Christopher M.
Ray, Navamita
Rogers, Michael Lloyd

Intended for: Report

Issued: 2017-09-19

Ristra Link Team

Ristra Milestone Review

Angela Herring, Gary Dilts, Charles Ferenbaugh, Rao Garimella, Eugene Kikinzon, Chris Malone, Navamita Ray, Mike Rogers

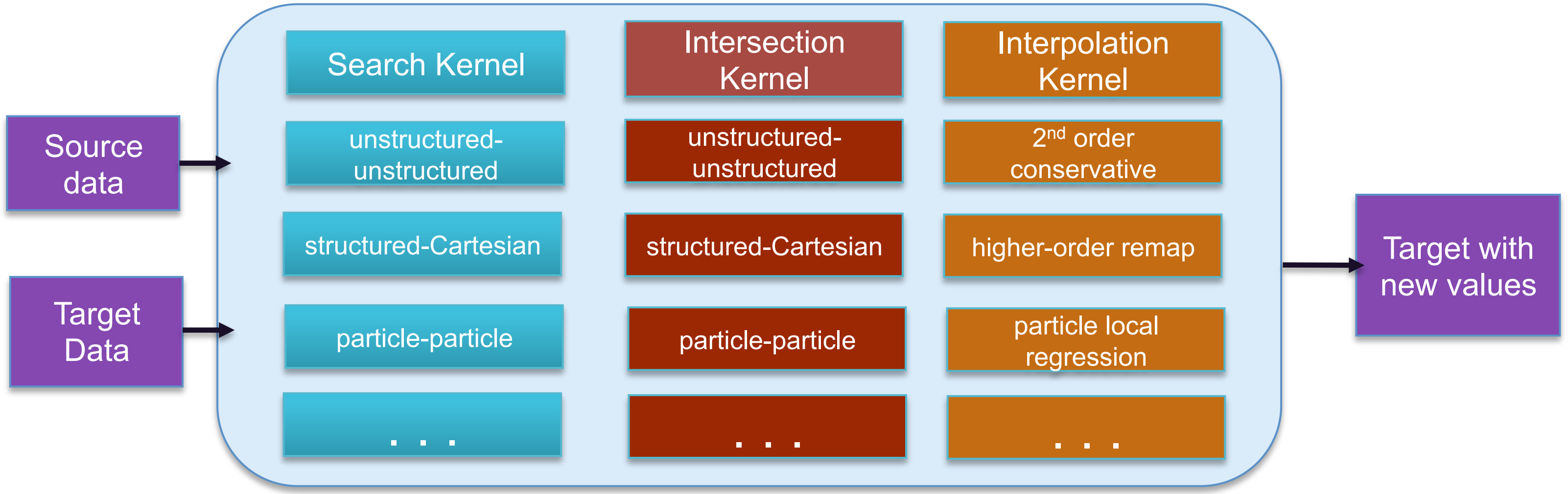
September 12, 2017

The Link Team works to make connections within and outside the Ristra project

- **Keeping ahead of Ristra needs:**
  - Portage update
  - Introduction to Tangram
- **Connection to the User Community: Engineering links**

Portage is Ristra's remap and link package

- **Targets a variety of link applications (inter-code and intra-code)**
- **Extensible (end-users encouraged to specialize algorithms)**
- **Specifically written for advanced architectures**
- **Designed for serial, on-node (shared-memory) parallelism and/or distributed-memory parallelism**
- **Independent of mesh/particle framework**



➢ *Search and intersect algorithms are physics agnostic, these should be adopted from Portage*

➢ *Portage will be the repository for interpolation schemes*

Portage is expanding its capabilities

Portage 1.0:

- 1st and 2nd order conservative 2/3-D cell and node-centered mesh-mesh remap
- Distributed and on-node parallelism
- Good weak and strong scaling

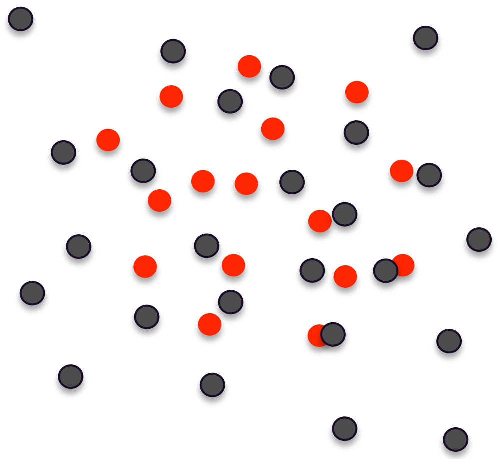
Portage 1.1:

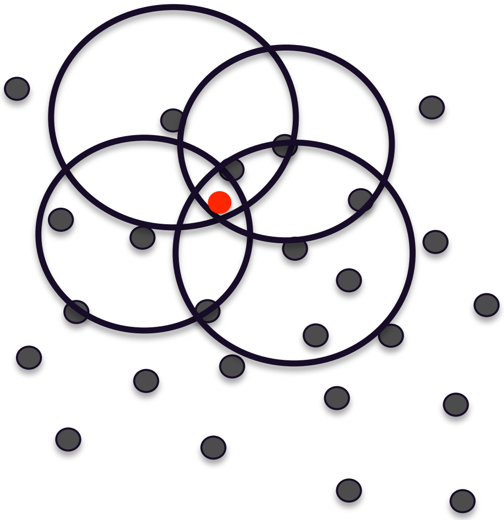
- 1st and 2nd order 2/3-D particle-particle remap
- Experimental 3rd order remap
- Compatible with Ristra toolkit 1.0 (including FleCSI)

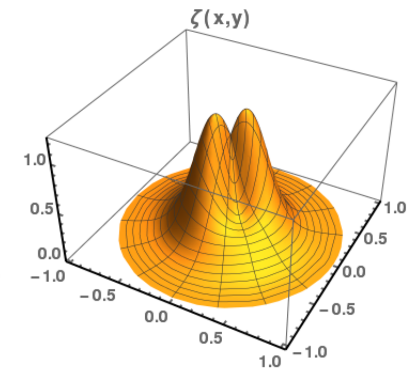
http://github.com/laristra/portage

Particle methods have a direct analog to mesh-based methods

Source
Target







<u>Search</u>

Find circles that cover target point

<u>Intersect</u>

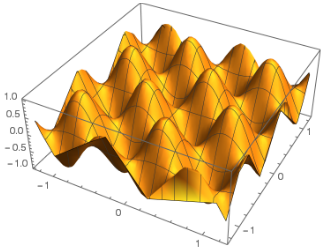
Find $\zeta(x,y)$

<u>Interpolate</u>

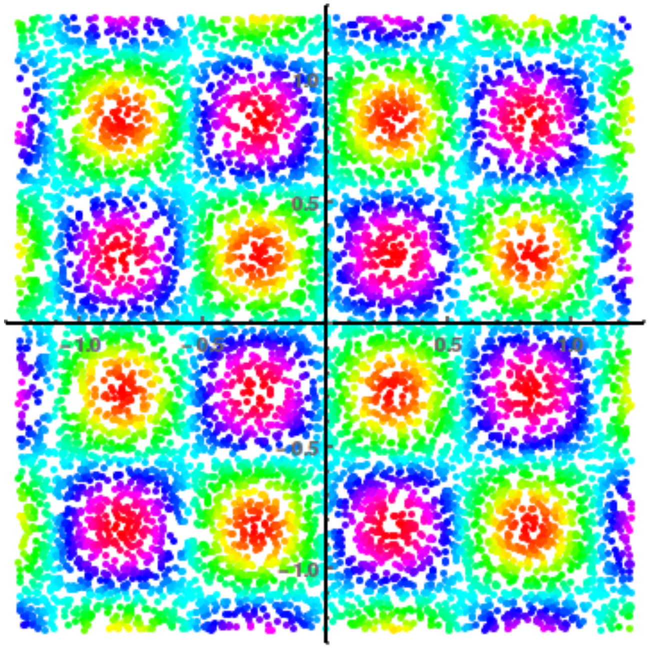
$u(x)=\sum_{i} \zeta(x,y_i)\, u_i$

Particle remap methods are flexible

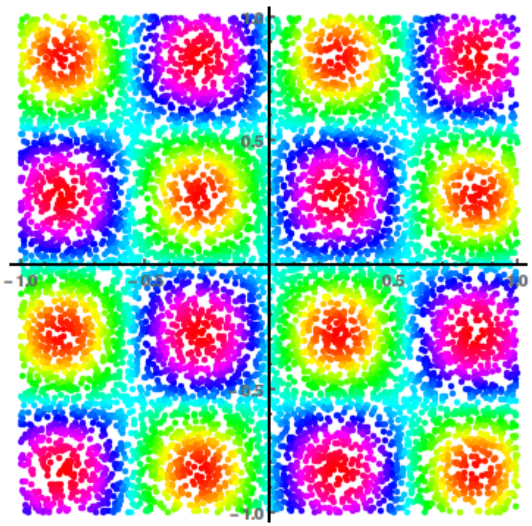
$\exp(-1.5r)\sin(1.8\pi x)\,sin(1.8\pi y)$



High-order particle methods can map data on random points to random points.



Source: 10000 random points

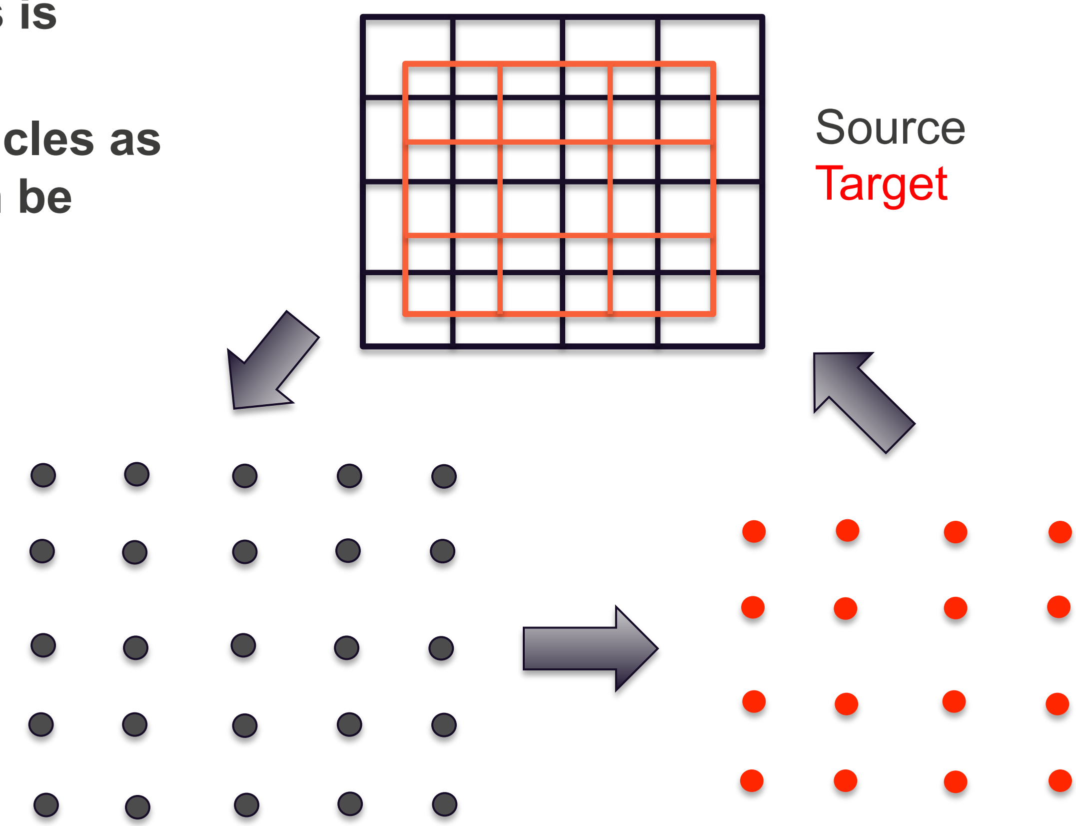


Target: 10000 random points

Particle remap is a step towards mesh<->particle remap

Using particles as an intermediary for remap may be faster than traditional mesh-based remaps

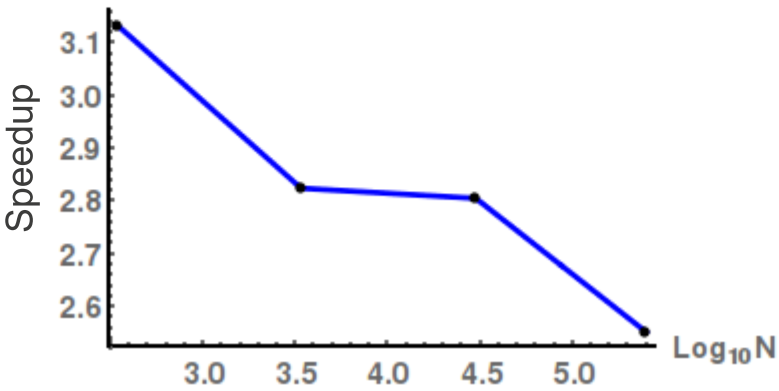
- Intersection of cells is expensive.
- Perhaps using particles as an intermediate can be faster?



Preliminary results are promising

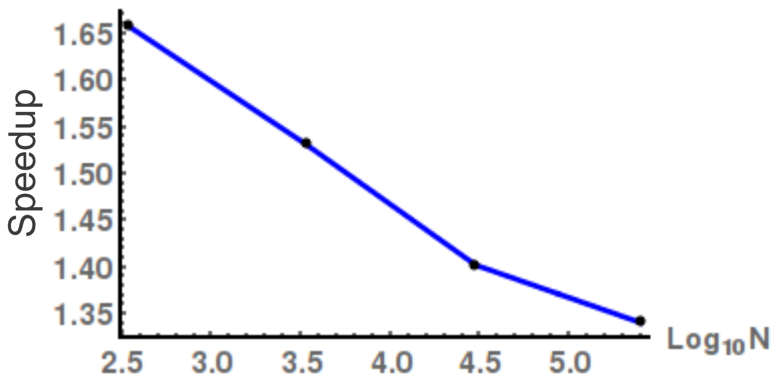
- Mesh faster at search, because neighbors are known
- Particles faster intersect and interpolate steps

First-order interpolation



Number of points/cells

Second-order interpolation

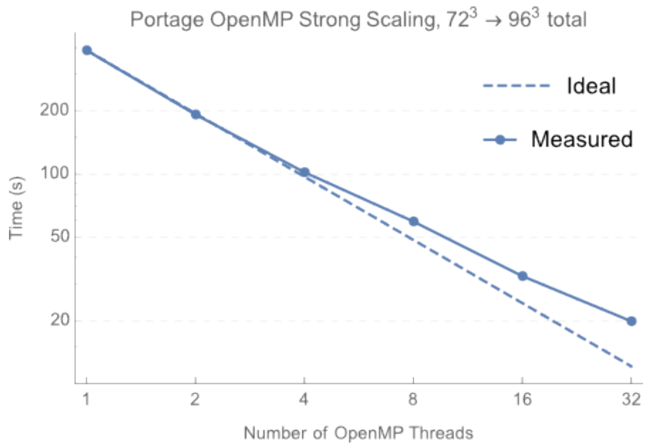


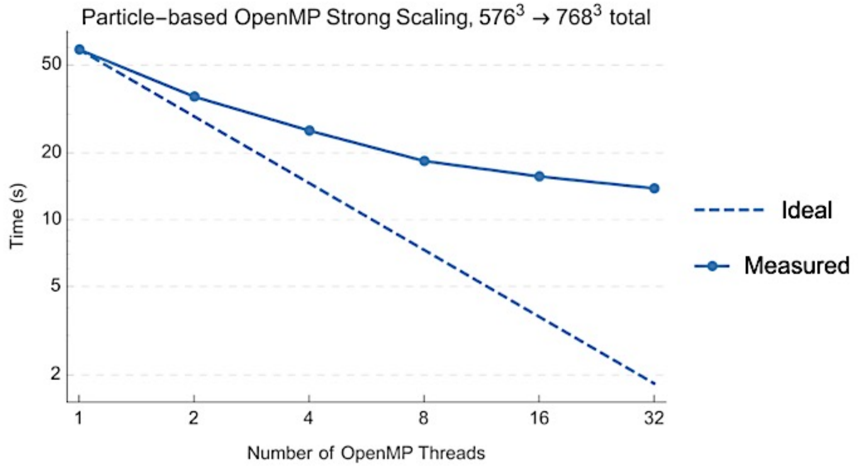
Number of points/cells

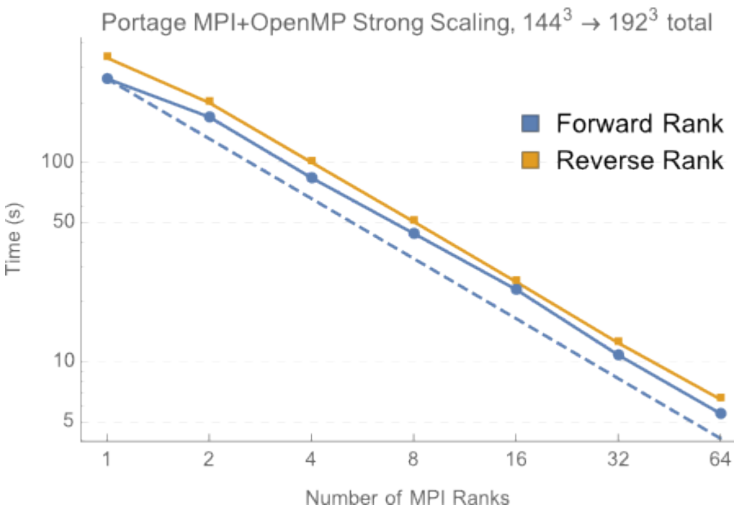
Many improvements can be made to our implementation

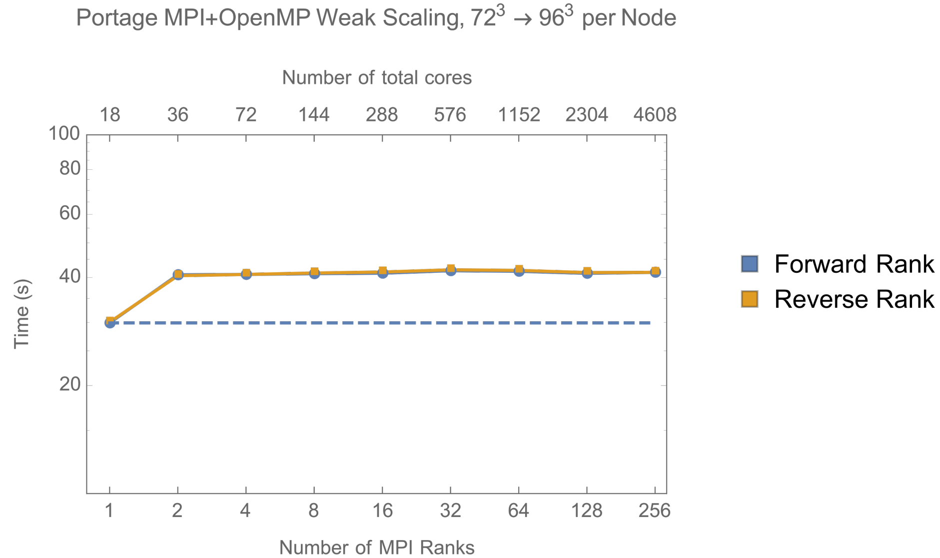
- **Current meshfree implementation is inefficient**
  - Much extraneous copying could be eliminated
  - Serial-only search can be made on-node parallel
  - Possibly faster LAPACK routine
- **Needs distributed MPI support**
- **Study performance for a broader range of problems**
  - Rotated box meshes
  - High aspect-ratio meshes
  - High source-target mesh size ratios
  - More intersections should favor meshfree even more

Portage's mesh-mesh remaps still scale well; more work needed on particle-particle remap





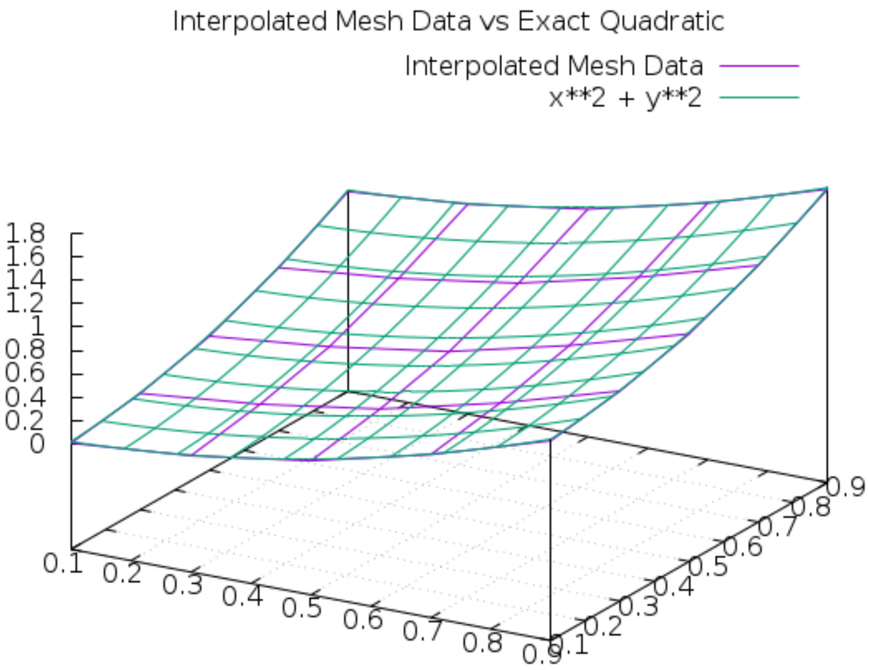




Portage can be extended by the user community

- ALE team targeting application which requires a 3rd-order remap to reduce dissipation errors to be comparable level as the Lagrange step (SGH)

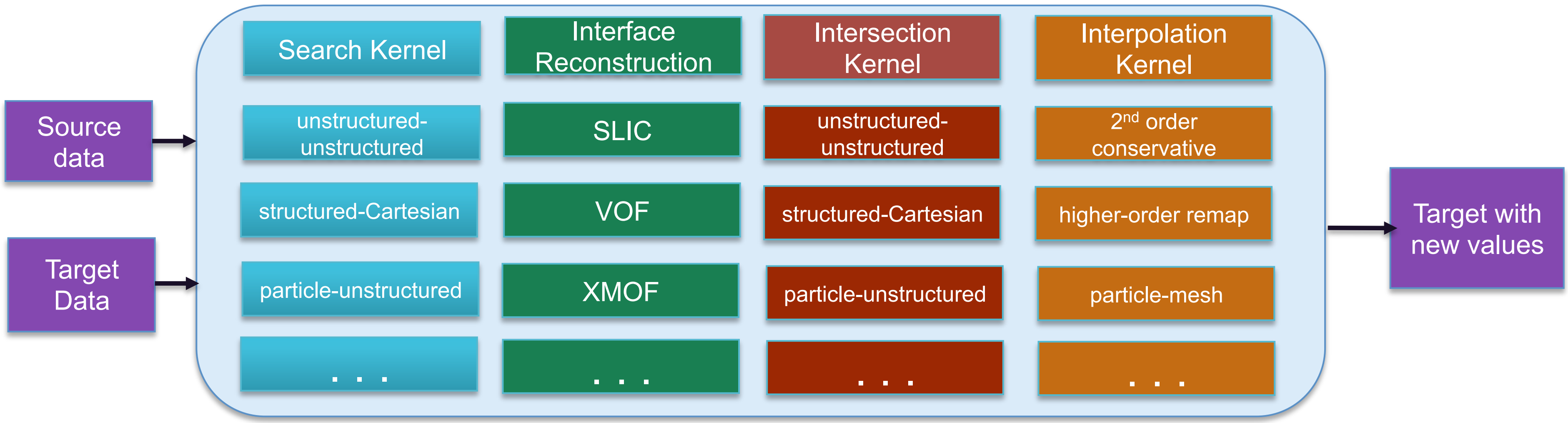
- Released with Portage 1.1 as an experimental feature



Portage Future work

- Multi-material remap
- Improved particle performance
- Verification of high-order remap
- Conservative particle remap
- Research: Particles as intermediary for remap
- Tighter integration with FleCSALE

Portage requires interface reconstruction for multi-material remap



- **Reconstruct pure material polygons on source mesh**
- **Intersect with target cell**
- **Return volume fractions on target mesh, material-wise properties on target mesh**

Interface reconstruction is useful in many other contexts

- **Advanced closure models for computing material state after Lagrangian deformation multi-material cells**
- **Computation of more accurate diffusion solution (one that avoids cell-level homogenization of material)**
- **Improved interface physics models:**
  - Material diffusion or atomic mixing across material interfaces
  - Eulerian slidelines, void creation and fracture along interfaces
  - Friction
- **Front propagation in evolving geometry problems such as HE burn**

Goal: Reuse as many capabilities as possible across applications

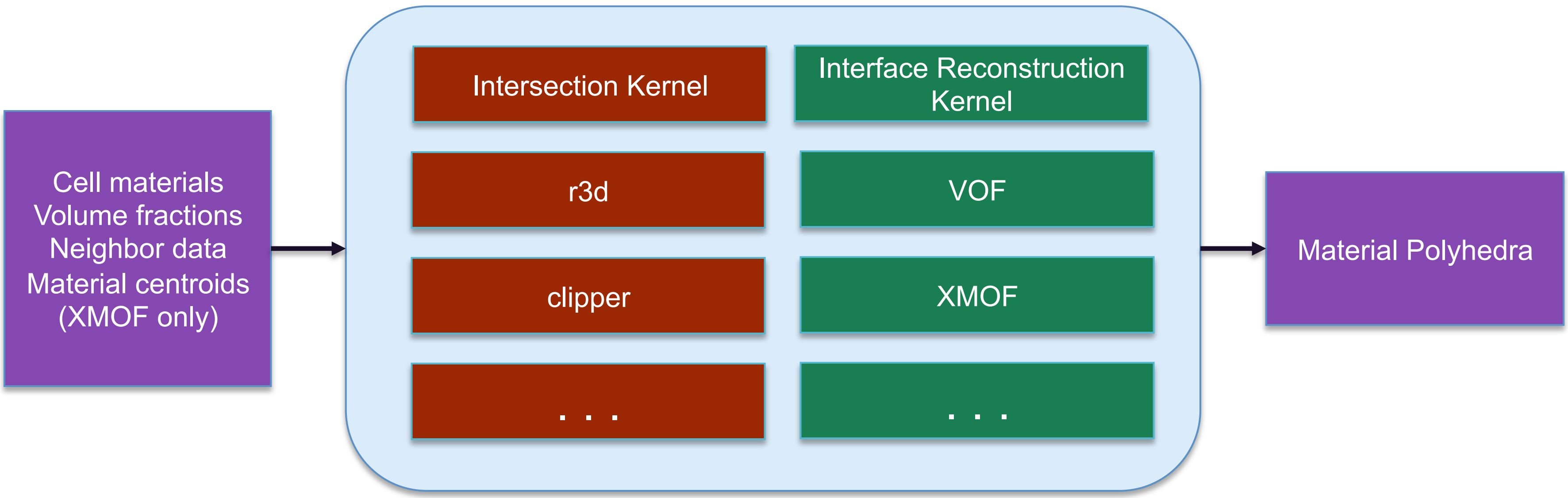
Tangram is an extensible, scalable interface reconstruction library

- **Collection of interface reconstruction algorithms**
- **Allows user to query results of interface reconstruction in uniform way**
- **Designed to be used in a variety of contexts**
- **Will leverage existing work**
  - Kikinzon/Shashkov (XMOF)
  - Flag IR, Telluride IR?
  - Others
  - R3D/R2D

Tangram will be the interface reconstruction package for Portage and FUEL.

Tangram takes design cues from Portage

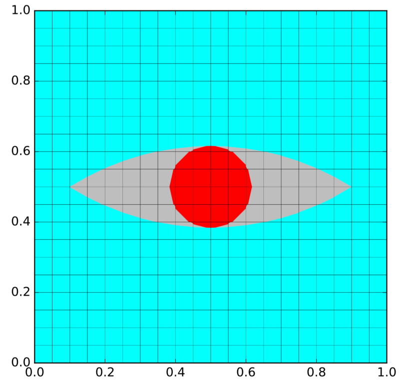
For each cell:

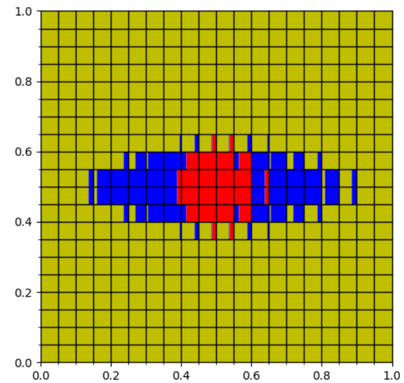


- **Leverages proven design principles:**
  - Modularity
  - Scalability
  - Extensibility
- **Reconstructs a single cell at a time**

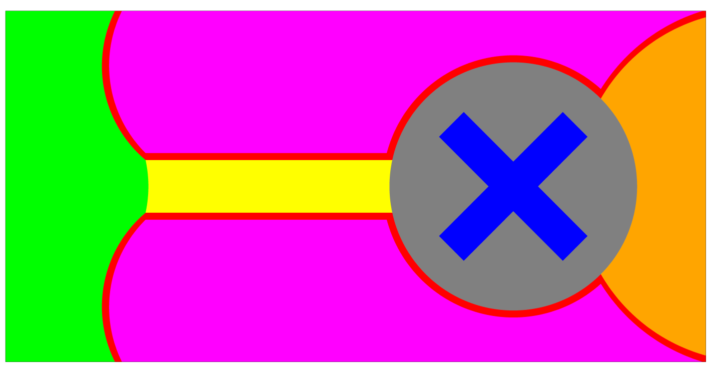
Tangram is off to a good start

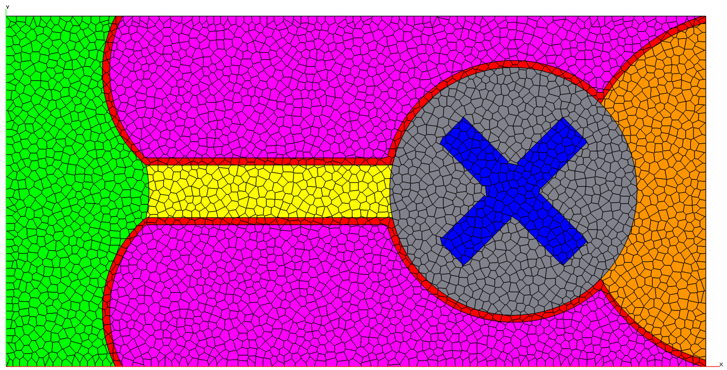
- **Skeleton code with simplified linear interface construction (SLIC) to demonstrate design**





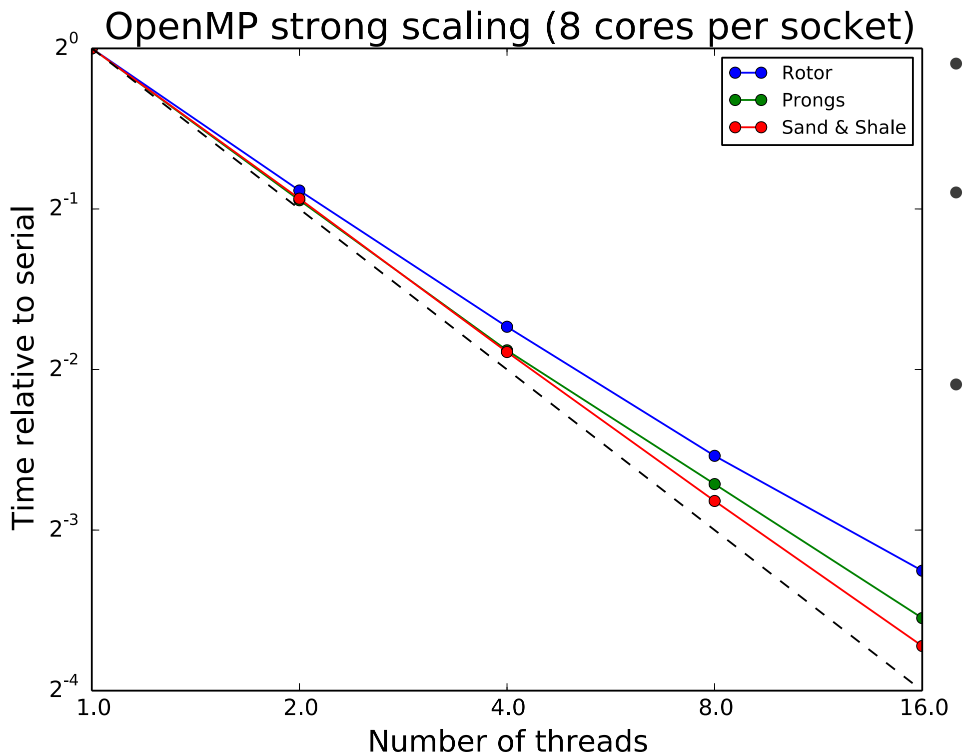
- **2-D XMOF now fully implemented**





Tangram scales well on-node

- **On-node interface reconstruction**
  - simple balancing: every threads gets the same number of multi-material cells
  - reconstruction cost per cell depends on the number of materials
  - reconstruction cost per material polygon depends on the geometry of the cut

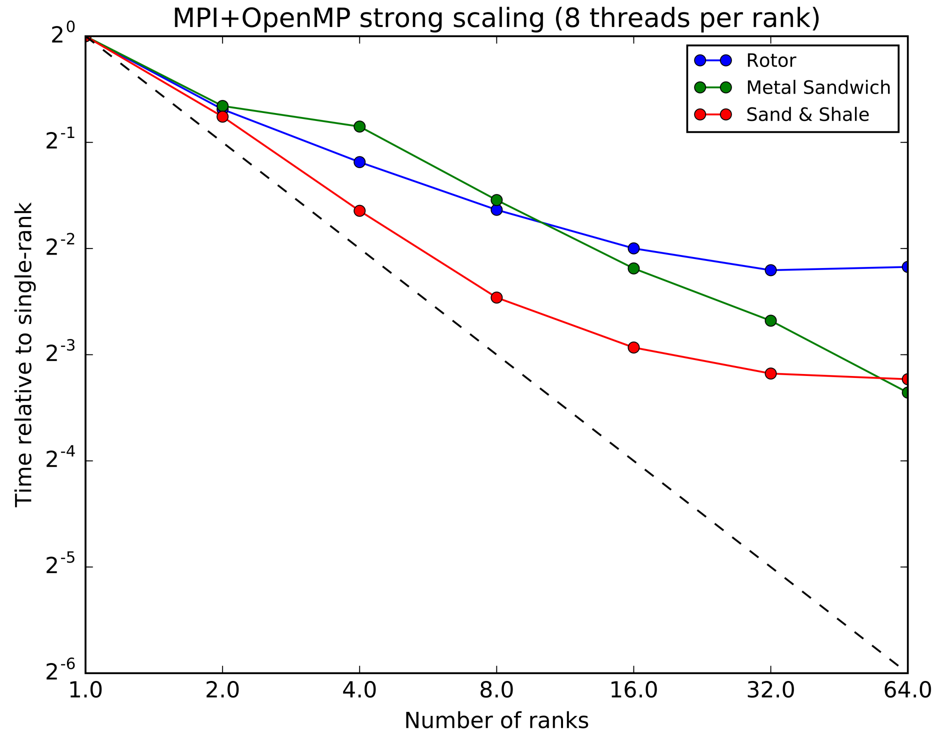


- **Red: 22922 multi-material cells**
  - two-material cells: 100%
- **Green: 23721 multi-material cell**
  - two-material cells: 98.2%
  - three-material cells: 1.8%
- **Blue: 4227 multi-material cells**
  - two-material cells: 85.61%
  - three-material cells: 14.34%
  - four-material cells: 0.05%

*A node comprises 2 x Eight-Core Intel Xeon [model E5-2670] @ 2.6 GHz
Tangram app was built using intel/17.0.1 and intel-mpi/2017.1 modules

Performance of Tangram depends strongly on mesh partitioning

- **Distributed interface reconstruction**
  - if performance of interface reconstruction is critical, partitioning should be optimized to level the number of multi-material cells on all ranks
- **Example of how this will look**
- **Thought needs to be given for integrated performance**



Tangram Future work

- Open source
- 3-D XMOF
- Integration with Portage
- Integration with FUEL

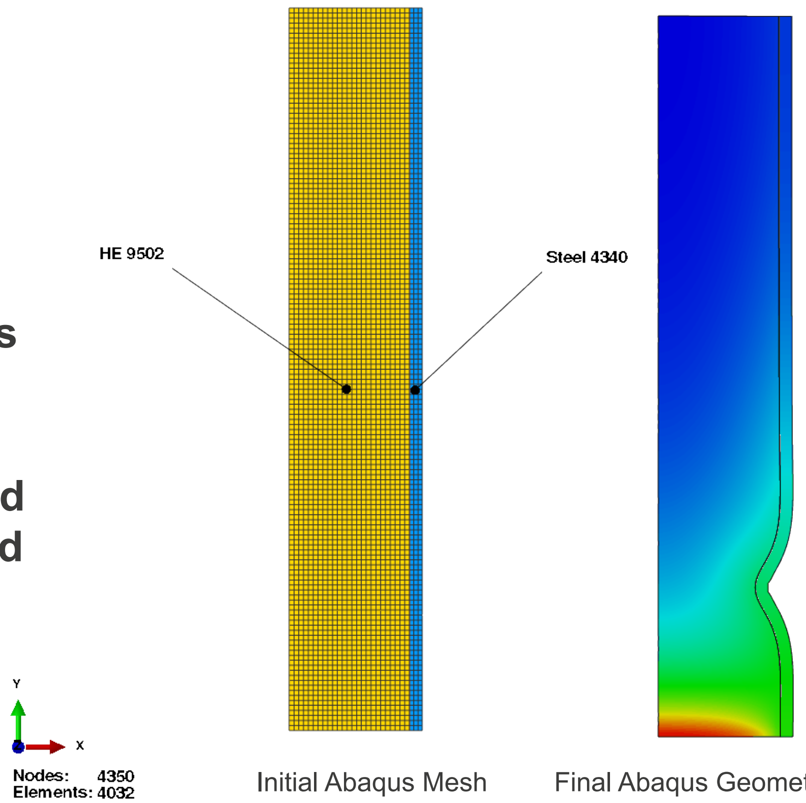
Ristra Partnerships with EAP & LAP have produced an automated physics field link

- Some simulations require the use of two codes , i.e. a ”linked” simulation
- Geometry link often sufficient to make an accurate assessment, but some require thermodynamic state/velocities
- EAP/LAP have a long standing interest in linking geometry and physics data from Abaqus
- Missing easy-to-reuse remapping capability—use Portage!
- Ingen is the logical place for this capability because it is the production meshing and geometry tool for EAP/LAP and the foundation on which CMF is being built.
- Portage integration enables Ingen to write native input files containing the new geometry and the field data for any code which utilizes the CMF framework or Ingen for its simulation setup

Test problem demonstrates the link process

- Abaqus simulation results database (.odb) imported directly into Ingen via the Abaqus Python interface
- Geometric models for both EAP and LAP generated in Ingen from the final Abaqus geometry
- Refined LAP hydro mesh created in Ingen
- Ingen calls Portage to generate link files with geometry and physics field data
- Once the workflow was established as part of NGC/EAP collaboration we tested export to and execution in FLAG and did a preliminary run

9502 encased in steel, heated and deformed in Abaqus

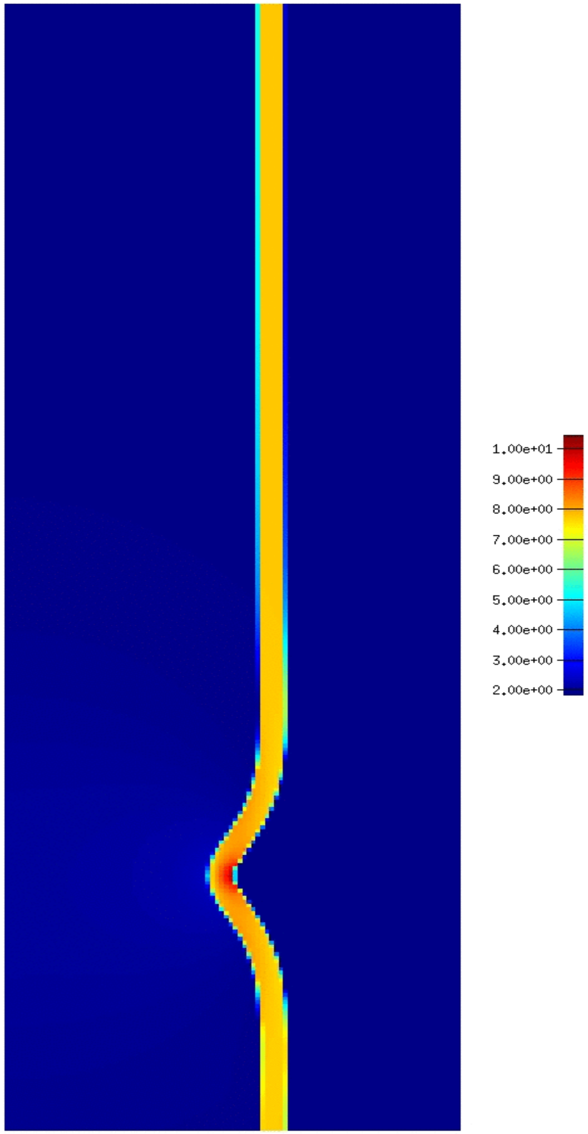


Initial Abaqus Mesh

Final Abaqus Geometry and Temperature

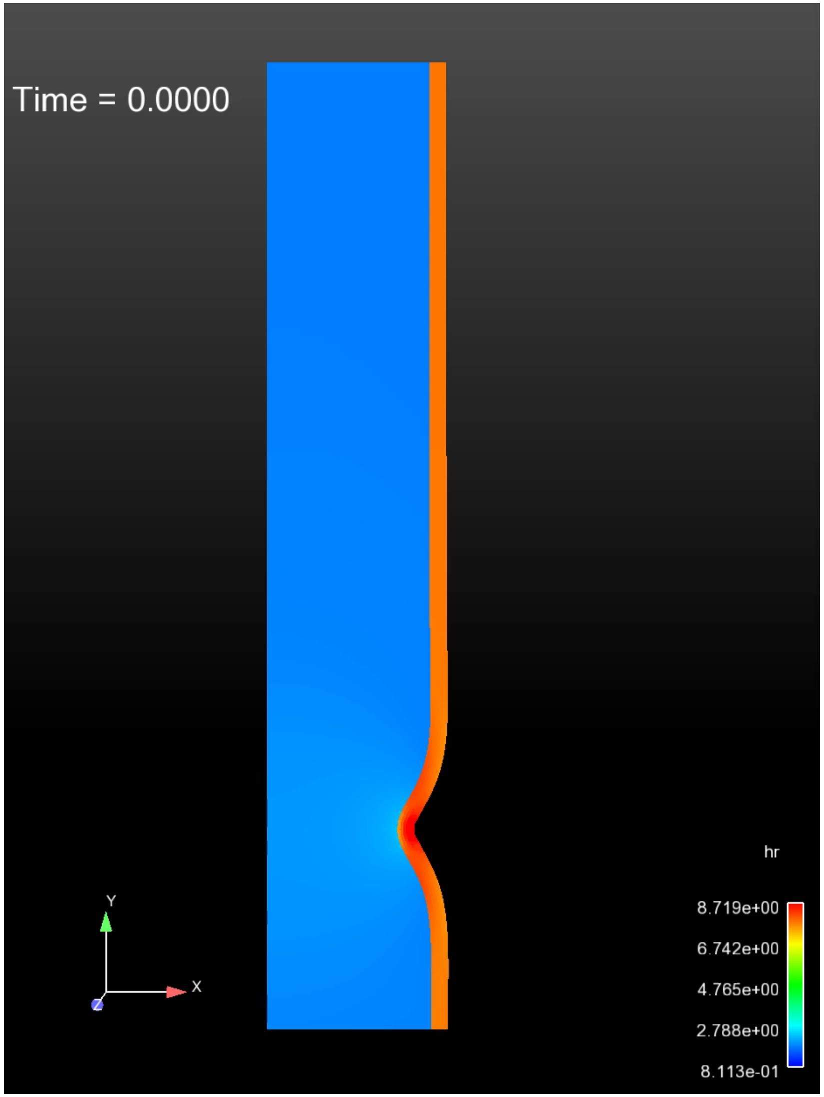
EAP

- **Model facts:**
  - Single detonator at 0, 0 at t=0 of xRAGE simulation
  - 2nd order remap from 28x144 to 62x304
  - Density, pressure and stress fields imported into xRage
  - SURF HE model



LAP

- **Model facts:**
  - Single detonator at 0, 0 at t=0 of FLAG simulation
  - 2nd order remap from 28x144 to 62x304
  - Lund HE model
  - Temperature, density linked
  - Movie shows density
- Ringing indicates need for more study



Broad collaborations have made this work possible

- Zach Medin, EAP
- Brian Jean, Lucy Frey, Guy McNamara, Tim Helton, Davis Herring, Rachel Ertl, Louie Long, Eli Feinberg, Laura Lang, SimTools
- Nick Denissen, Brandon Smith, LAP
- Rajeev Erramilli, Princeton
- Peter Crossman, Gordon College